情報処理概論期末試験

問１. 次の語句を簡単に説明しなさい．(各 2.5 点) (具体例によって説明しても良い．)

(1) bit
(2) byte
(3) 識別子
(4) プログラミング言語
(5) 高級言語／低級言語
(6) java 言語
(7) オブジェクト
(8) クラス
(9) インスタンス
(10) フィールド
(11) メソッド
(12) 単項演算子
(13) ホワイトスペース
(14) インクリメント
(15) 2 の補数
(16) CPU
(17) フェッチ
(18) デコード
(19) ストアードプログラム型計算機
(20) UML

問２. バブルソートを実行する下のプログラムについて次の問いに答えなさい。

(1) このプログラムを実行したときに、表示されるものを書きなさい。(各 10 点)

(2) 行番号が付加されている各行の内容を行番号ごとに説明しなさい．(各 20 点)

(3) このプログラムは正しく動作するが，変数とプログラムの多少の付加・修正で，さらに効率的なプログラムになる．そのためにはどこをどのようにすれば良いか理由とともに示しなさい．(各 20 点) (最終的な結果は同じであるか，途中の表示は異なる．)

```
 1: class Kimatsu {
 2:     public static void main(String args[]) {
 3:         BSort a = new BSort();
 4:         a.bSort();
 5:     }
 6: }
 7: class BSort {
 8:     int  nData, data[];
 9:     BSort() {
10:         nData  = 4;
11:         data  = new int[nData];
12:         data[0] = 1; data[1] = 3; data[2] = 4; data[3] = 2;
13:     }
14:     void bSort() {
15:         int loop = 0, tmp;
16:         while(true) {
17:             printData();
18:             if (loop < 0) loop = 0;
19:             if (loop == nData - 1) break;
20:             if (data[loop + 1] < data[loop]) {
21:                 tmp = data[loop];
22:                 data[loop] = data[loop + 1];
23:                 data[loop + 1] = tmp;
24:                 loop -= 1;
25:             }
26:             else loop++;
27:         }
28:     }
        void printData() {
            int loopPrint;
            for (loopPrint = 0 ; loopPrint < nData ; loopPrint++) {
                System.out.print("data[" + loopPrint+  "] = " + data[loopPrint] + "  ");
            }
            System.out.println();
        }
    }
```

学科・類　　学籍番号　　-　　-　　氏名

問1

(1)

(2)

(3)

(4)

(5)

(6)

(7)

(8)

(9)

(10)

(11)

(12)

(13)

(14)

(15)

(16)

(17)

(18)

(19)

(20)

問2

(1)

(2)

(3)

問1

(1) 情報量の単位で，1bit は 0 または 1 などの二者択一の情報量を表す.

(2) 1byte は 8bit の情報量を意味する．(昔は，1word の意味にも使われていたが特に解答する必要はない。)

(3) 変数やメソッドなどを識別するためにつける名前.

(4) コンピュータに行わせたい仕事を記述するための言語.

(5) 低級言語は計算機が直接実行できる機械語に近い言語。高級言語は人間が使用している言語に近づけた言語。

(6) サンマイクロシステムが作ったオブジェクト指向言語で，バーチャルマシンによって様々なコンピュータでも実行できる特色を持つ.

(7) 情報を処理するために，その対象の属性と操作をまとめたもの．既存の手続き型では，この両者が別れて記述されていたが，オブジェクトによって統合的に記述できるようになった.

(8) オブジェクトの設計図．属性の宣言や操作を具体的記述を行う.

(9) 「具体化したもの」．あるクラスのインスタンスがオブジェクト.

(10) Java などにおけるオブジェクトの属性のこと．(オブジェクトの内部の変数)

(11) Java になどおけるオブジェクトの操作のこと.

(12) オペランド (演算の対象) を 1 つだけとる演算子のこと．例えば，インクリメント演算子 (++) などがある.

(13) Java プログラムの区切を示すためのもので，スペース,、改行，タブのこと

(14) あるものの 1 つ分を加算すること．++のこと.

(15) 計算機内で，負の数を表すために使われる．符号つき 16bit の整数 (先頭 1 ビットが符号) においては，0x10000 にその負の数を加算して得られたもの。

(16) セントラル・プロセッシング・ユニットの略で，プログラムの指示に従って実際にデータ処理や計算を行うところ.

(17) CPU がプログラムを取ってくる動作.

(18) CPU が取ってきたプログラムを解釈する動作.

(19) プログラムをデータとほぼ同様にメモリー上に置いて，プログラムを取り出しながら処理する計算機.

(19) ユニファイド・モデリング・ランゲージ．様々なものをオブジェクト指向によってモデリングして記述するためのもの。クラス図，シーケンス図等がある.

問2

(1)

```
data[0] = 1  data[1] = 3  data[2] = 4  data[3] = 2
data[0] = 1  data[1] = 3  data[2] = 4  data[3] = 2
data[0] = 1  data[1] = 3  data[2] = 4  data[3] = 2
data[0] = 1  data[1] = 3  data[2] = 2  data[3] = 4
data[0] = 1  data[1] = 2  data[2] = 3  data[3] = 4
data[0] = 1  data[1] = 2  data[2] = 3  data[3] = 4
data[0] = 1  data[1] = 2  data[2] = 3  data[3] = 4
data[0] = 1  data[1] = 2  data[2] = 3  data[3] = 4
```

(2)

1: クラス Kimatsu の宣言を開始している.

2: はじめに実行するためのメソッド main の宣言を開始している.

3: クラス BSort のオブジェクトをコンストラクトして，それを変数 a に格納している.

4: BSort のオブジェクト a のメソッド bSort を実行している.

5: メソッド main の宣言を終了している．

6: クラス Kimatsu の宣言を終了している．

7: クラス BSort の宣言を開始している．

8: クラス BSort のフィールドとして，整数型の変数 nData と整数型配列 data を宣言している．

9: クラス BSort のコンストラクタの宣言を開始している．

10: 整数型のフィールド nData に 4 を代入している．

11: 整数型配列のフィールド data のために，nData 個分の領域を確保して，割り当てている．

12: 整数型配列のフィールド data に対して順番に，1, 3, 4, 2 を代入している．

13: BSort のコンストラクタの宣言を終了している．

14: メソッド bSort の宣言を開始している．

15: 整数型の変数 loop と tmp を宣言して，loop に 0 を代入している．

16: 次の中括弧で示される部分に対する無限ループを示している．(ループ内部の break 文によってループが終了する．)

17: 配列 data の内容を表示するメソッド printData を実行することを示している．

18: loop がマイナスの時は調べることがないので，0 から調べる．loop がマイナスになる理由は，loop が 0 の時に、整列順番が逆でデータを入れ替えた後にその 1 つ前を調べようとするため．

19: loop が nData - 1 になれば，全てのデータの整列が終了したことになるので，break でループを終了する．

20: 配列の loop 番目の値とその次 (loop + 1) 番目の値を比較する．もし後者の方が大きければ入れ替える必要があるので，21〜24 の処理を行う．そうでないときは，26 の処理を行う．

21: 21〜23 は，両者のデータの入れ替えである．まず，loop 番目の値を tmp に代入することを示している．

22: (loop + 1) 番目の値を loop 番目に代入することを示している．

23: tmp の値を (loop + 1) 番目に代入することを示している．

24: 1 つ前のデータと比較するために，loop から 1 を引くことを示している．

25: 20 の if 文の条件が成立したときの実行の終りを示している．

26: 20 の検査でデータの並び順が正しかったので，次を調べるために loop に 1 を加算することを示している．

27: 16 からはじまるループの処理の終りを示している．

28: メソッド bSort の宣言の終りを示している．

(3)

バブルソートでは，整列順序になっていないデータを入れ替えた後で，その 1 つ前のデータを調べに行く．このようにして入れ替えていった後に調べた結果，順序が正しかったとき，入れ替えはじめた次のデータまで戻って順序が正しいかどうか調べれば良い．上のプログラムの無駄なところは，その点まで戻るときに，データの整列関係を 1 つずつ調べていることである．

下のプログラムでは，その入れ替えはじめたところを loopMax で記憶しておき，順序が正しかったときには，その次の値に loop を設定することによって，その点まで一度に戻ることができる。

(解答は修正点だけ指摘すれば良い．)

```
  void bSort() {
      int loop = 0, loopMax = 0, tmp;         //修正
      while(true) {
           printData();
          if (loop < 0) loop = ++loopMax;     //修正
          if (loop == nData - 1) break;
          if (data[loop + 1] < data[loop]) {
              tmp = data[loop];
              data[loop] = data[loop + 1];
              data[loop + 1] = tmp;
              loop -= 1;
          }
          else loop = ++loopMax;                //修正
      }
  }
```

(ちなみに，このプログラムによる出力は，以下のようになる．)

```
data[0] = 1  data[1] = 3  data[2] = 4  data[3] = 2
data[0] = 1  data[1] = 3  data[2] = 4  data[3] = 2
data[0] = 1  data[1] = 3  data[2] = 4  data[3] = 2
data[0] = 1  data[1] = 3  data[2] = 2  data[3] = 4
data[0] = 1  data[1] = 2  data[2] = 3  data[3] = 4
data[0] = 1  data[1] = 2  data[2] = 3  data[3] = 4
```